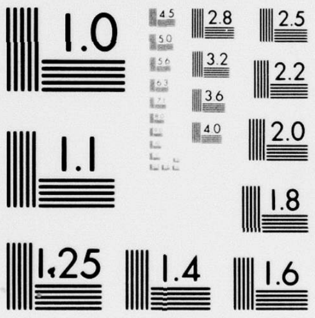
AD-A078 463   CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF ELECTRICA--ETC F/G 9/2
AUTOMATED SYNTHESIS OF DIGITAL HARDWARE MODULES: SIMULATION AND--ETC(U)
OCT 79 A C PARKER , D E THOMAS   DAAG29-73-G-0070

UNCLASSIFIED   ARO-13670.16-EL   NL

| OF |
ADA
078463

END
DATE
FILMED
1 -80
DDC



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(18) ARO (19) 13670.16-EL

(12)

LEVEL II

ADA078463

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (6)	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER (9)
4. TITLE (and Subtitle) AUTOMATED SYNTHESIS OF DIGITAL HARDWARE MODULES; SIMULATION AND VERIFICATION OF INTERCONNECTIONS		5. TYPE OF REPORT & PERIOD COVERED FINAL rept. June 1, 1978 - Aug. 31, 1979
		6. PERFORMING ORG. REPORT NUMBER 1 Jun 78-31 Aug 79,
7. AUTHOR(s) (10) Alice C. Parker, Donald E. Thomas	(15)	8. CONTRACT OR GRANT NUMBER(s) DAAG29-73-G-0070
9. PERFORMING ORGANIZATION NAME AND ADDRESS Electrical Engineering Department Carnegie-Mellon University; Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS (11) 31		12. REPORT DATE ~~October 31, 1979~~
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (12) 17		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official department of the Army position, policy, or decision unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Simulation, Synthesis, Digital-system design, Design automation, Verification, Input-output, Buses, Microprogramming, Control design, Design optimization, Hardware-descriptive language.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The research described in this final report involves to basic areas: automated design and optimization of digital hardware, and simulation and verification of interconnections between digital system modules. Both of these are part of the CMU-DA (Design Automation) project (formerly called the RT-CAD project). We have investigated strategies for computer-aided optimization of the control part of digital systems hardware by developing a discrete optimization technique and a strategy for applying this technique. In addition, we have developed a
(over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

403 445

79 11 27 011

20.
design aid for computer interconnection evaluation by simulation techniques. Finally, we have begun to design a system to formally verify that a description of interconnections is functionally correct.

AUTOMATED SYNTHESIS OF DIGITAL HARDWARE MODULES; SIMULATION AND VERIFICATION OF INTERCONNECTIONS

FINAL REPORT

Alice C. Parker

October 31, 1979

U.S. Army Research Office

Grant Number DAAG29-78-G-0070

Department of Electrical Engineering

Carnegie-Mellon University

Automated Synthesis of Digital Hardware Modules; Simulation and Verification of Interconnections

ABSTRACT

The research described in this final report involves two basic areas: automated design and optimization of digital hardware, and simulation and verification of interconnections between digital system modules. Both of these are part of the CMU-DA (Design Automation) project (formerly called the RT-CAD project). We have investigated strategies for computer-aided optimization of the control part of digital systems hardware by developing a discrete optimization technique and a strategy for applying this technique. In addition, we have developed a design aid for computer interconnection evaluation by simulation techniques. Finally, we have begun to design a system to formally verify that a description of interconnections is functionally correct.

1.0 Introduction and Research Objectives

VLSI designers are going to be faced with two problems - optimum design of the VLSI chips themselves and also optimum interconnection strategies for these chips. In order to achieve the cost-effective, reliable, easy-to-use and/or high-performance systems desired by the user, the chip and system designers will require more powerful design aids than those currently available.

In a recent workshop on the science of design, sponsored by the Office of Naval Research [11], a number of research areas were discussed. Areas where the participants felt more research was needed included design verification, techniques for hardware synthesis and I/O considerations. In specific, the following points were raised:

- The need for models of parallel structures.
- The need for intersystem compatibility.
- The necessity of developing models that consider the communication demands of algorithms.
- The need for design aids that provide for optimization at the system level.
- The need for techniques for mapping algorithms to standard building blocks or new achitectures.
- The need to quantify the architectural selection process.
- The need to implement design verificaton throughout the entire design process.

It has been the intent of the research described here to produce results compatible with the above needs.

The research described here is a continuation and outgrowth of current design, simulation and design automation research underway at Carnegie-Mellon University. The ultimate goal of this entire research area is an advancement of the state of knowledge of design automation, exemplified by a comprehensive package of design aids for top-down synthesis, simulation and design verification. The package is intended to provide the designer with the ability to explore many alternative designs and their behavior in his search for the optimal[1] design. In addition, the designer should be able to easily verify correctness of the resulting designs and to simulate the designs on a number of levels.

[1] we use optimal here to indicate the design solution which meets the cost, speed, reliability, power and other criteria better than any other design solution which can be found.

With these general goals in mind, we have continued our simulation and design automation efforts and begun design verification research at several levels. In particular, the outcome of this research has been:

- A simulation tool for evaluating bus and I/O designs similar to the MCF [5] architecture evaluation. This simulator will be used to evaluate a proposed ANSI bus standard [4]. (Evaluation of proposed MCF buses [7] is another possible application of these techniques.)
- Preliminary research into a formal technique for verifying correctness of an I/O or bus hardware description.
- Basic research into multiple-criterion optimization of the control hardware of digital designs.

2.0 Present Research Status of the Entire CMU-DA Project[2]

Present research at Carnegie-Mellon has been focusing on design synthesis, multilevel simulations and formal I/O and bus descriptions. The synthesis research is the longest running and largest of the two projects. A brief summary of the research will be presented here, along with a system block diagram showing status of each project. More comprehensive discussions of the effort can be found in [12, 16, 18, 15, 21].

2.1 The Present CMU Design Automation System

<u>CMU-DA Overview</u> [19]

The ultimate goal of the CMU-DA project is to provide a technology-relative, structured-design aid to help the hardware designer explore a larger number of possible design implementations. Inputs to the system are a behavioral description of the system to be designed, an objective function which specifies the user's optimization criteria, and a library specifying the hardware components available to the design system. The componets of the CMU-DA system are shown in Figure 1 and discussed below.

The CMU-DA system differs from other design automation systems in that it operates from a behavioral specification of a piece of hardware that does not necessarily reflect its internal structure. More and more structural detail is frozen at each level of the CMU-DA system until a complete hardware specification is obtained, with the most influential design parameters being examined first in order to cut down the design search space. The functions of the

[2]This project has been funded by a number of government and industrial grants.

design system components which bind these implementation decisions are described below.

GLOBAL OPTIMIZER. The global optimizer applies high-level transformations to a design's behavioral representation after translating it from ISPS notation [6] to an abstract design representation called the value trace [18, 14]. The transformations have a significant impact on the cost, performance, and other parameters of the designs to which they are applied.

DESIGN STYLE SELECTOR. By considering the various module sets that can be used (e.g., TTL vs. a microprocessor), the design contraints imposed (e.g., pipeline data flow), the design style selector decides on the specific style of design to be employed (e.g., bit-slice microprocessor, MOS microprocessor, SSI/MSI logic). Earlier work [21] shows this to be an influential decision in terms of cost and speed tradeoffs.

PARTITIONER. The partitioner groups operations from the abstract design representation into tentative control steps. This effectively binds the control flow for the design. Tradeoffs between the data and control parts are made at this level.

DATA/MEMORY (DM) ALLOCATOR. The function of the DM allocators is to decide the number and type of data operators, multiplexers, and registers needed to implement the data part of the design, and to group operations from the abstract design into specific control steps. The output of the allocator is a data path graph whose nodes are elements such as adders or registers; the edges represent actual circuit interconnections. Recent results indicate that a non-optimal allocator can produce data paths with costs well within 50% in excess of human designs.

CONTROL ALLOCATOR. The control allocators generate a sequential state machine to control the data paths produced by the DM allocator. Each control allocator is to design the control unit around a particular control philosophy such as microprogramming, programmed logic arrays, random logic, etc. The output of the control allocator is a control path graph which is an abstract representation of the control hardware.

MODULE BINDER. The module binder selects physical modules from the module set library to implement a design's data and control path graphs. The library contains available components and may be freely updated, partially providing the technology-relative aspects of the CMU-DA system.

PHYSICAL LAYOUT PROCESSOR. This component partitions the system into printed circuit boards or chips, decides on the placement of components, routes interconnections, and prepares engineering documentation.

2.2 I/O and Bus Research

I/O and bus specification, verification, and simulation are currently being researched. The formal specification language, SLIDE, has now been completed, and a compiler with substantial diagnostic capabilities has been built and tested [23, 25].

3.0 Summary of Research Progress and Results

Progress has been make in the areas of control hardware synthesis and optimization, simulation of buses and other interconnection strategies, and formal verification of interconnection synchronization mechanisms. Each of these areas will now be discussed briefly.

Control Hardware Synthesis and Optimization

Given a completed data-path design and the corresponding control signals which describe the usage of the data paths, the control hardware must be automatically designed. Each synthesis program for the control (the control allocator) implements a different design style; we have selected a microprogrammed control mechanism as our example design style. It would be tempting to have assumed a canonical microprogrammed controller for this purpose, with horizontal micro-instructions and a simple two-way branch-on-condition mechanism. Then, microcode could have been generated for a known control mechanism. However, if we are to generate designs which meet the user's cost/speed constraints, it is important to consider a range of tradeoffs in control design and to develop a methodology for applying these tradeoffs.

Fortunately, many controller design tradeoffs are well understood. Agerwala [1] summarizes some of these, including minimization of the state count and microinstruction width, and detection of parllelism in straight-line microcode.

Two fundamental optimizations which have been analyzed are microoperation packing and reduction of microinstruction word width. The packing optimization (word optimization) can be formulated as a microcode optimization which is NP complete [10]. The encoding for width reduction (bit optimization) can be formulated as a set partitioning problem which is also computationally infeasible to perform optimally.

However, a near-optimal linear algorithm for packing of microoperations into microinstructions has been found and reported on recently [13], based on an algorithm published by Dasgupta and Tartar [8]. Unfortunately microinstruction packing and bit encoding into microwords cannot be performed independently because the degree to which the former is performed affects the degree to which the latter can be performed, and vice-versa. Thus a new algorithm has had to be researched.

Our progress in this area has involved the representation of the control graph for a given digital design, and the optimization/synthesis algorithm which manipulates the graph. At present, a representation for the control graph has been derived, as shown in Figure 2. Information contained in this graph includes the microoperations (micro-ops) which control the data part, the data-part modules which are controlled, and the necessary time-precedence relationships between the micro-ops. The information in the graph is a combination of information generated during the module-binding process and data accessed by the control-synthesis routine, from the module data base, as it builds the control graph. The structure of the graph is important because of the potential complexity of the optimization/synthesis algorithm which operates on the graph. The software routine which generates this control graph has been written and debugged.

Once the graph exists, it must be mapped into a microprogram and the hardware for the control store must be synthesized. The optimization/synthesis algorithm which performs this function is unique. Its distinctive features are:

- It performs word and bit optimizations concurrently.
- It adapts to cost/speed constraints input by the user by balancing between word and bit optimizations.
- It is computationally inexpensive.

The algorithm determines both which control step (in time) a given microoperation belongs to, and also how to encode the microinstructions. Thus it performs both word and bit optimizations, as defined by Agerwala [1], concurrently. It is obvious that these two optimizations are competing; the former attempts to pack micro-ops into microoperations so as to minimize the number of control steps and the latter attempts to encode micro-ops into microinstruction fields, excluding the possibility of their concurrency. Thus, the algorithm adapts by performing word optimization only to the point that the word width becomes too large to meet the cost constraints.

At this point, there are two solutions: if the speed constraint forces the attempted concurrency, the micro-op packing will continue. If, however, the speed constraints will be met even if additional control steps are introduced or the cost constraint is given higher priority by the user, the algorithm creates a new microinstruction format and a format selection field. Then, microoperations start being packed into the new format. The microcode becomes more vertical in nature.

This optimization/synthesis algorithm operates by examining relationships between the required microoperations and by combining this information with frequency-of-execution information to produce attraction weights between pairs of microoperations. For instance, if two microinstructions occur frequently in the control graph in positions which could be partitioned into the same control step, their attraction weight will be high. (Control step partitions are determined by the longest path through the most parallel implementation of the graph). The algorithm uses the attraction weights to determine candidate microoperations to be packed into microinstructions. Because the attraction weights are based on probabilities, the remaining attraction weights will need to be recomputed every time a binding to control steps (packing) is performed.

Concurrently, bit optimization is performed iteratively as the concurrency of microoperations is determined. Thus, the bit optimization operates only on microoperations which are already determined not to occur concurrenly. Bit optimization is prformed by overlaying nonconcurrent control signals used for path and function selection into the same fields in the microword. These signals are called "SELECT" signals. Signals which control register-transfers, "EVOKE" signals, which occur concurrently are given individual bits in a miroinstruction format. If more than one microinstruction format has had to be created, format selection bits determine the interpretation of the SELECT and EVOKE fields. This technique allows microcode packing without encoding [15].

The algorithm is computationally inexpensive, and although the complexity of execution is data-dependent, it appears that worst-case execution can be performed in polynomial time. The optimization software is running, and preliminary results indicate near-optimal microcode with capability to adapt to user cost/speed constraints. Publication of this algorithm will occur by early 1980.

Simulation of Interconnections

The outcome of the I/O simulation research is a simulator designed to accept and simulate any digital system interconnection scheme. This scheme must be described formally in SLIDE, a hardware descriptive language [23] and compiled and linked to the simulator. Simulations are behavioral, with some timing considerations. The core of the simulator is a multi-purpose simulator [9] designed for use on other simulation problems as well. The I/O simulator is further described in [2] attached as Appendix A and has performed one example simulation. This example consisted of a UNIBUS[3]-like structure interfaced to a single-wire data-link with

[3]UNIBUS is a registered trademark of Digital Equipment Corporation.

an SDLC[4]-like protocol. The SLIDE compiler was written in BLISS, the code generator in SIMULA, and the simulator in SIMULA.

Formal Verification of Synchronous Mechanisms

Preliminary work has begun in the area of formal verification of hardware synchronization mchanisms [24]. There is an analogy between this work and protocol verification [26, 20]. This research has involved modifying the SLIDE syntax to produce V-SLIDE, a verifiable form of SLIDE. The modifications allow encapsulation of the synchronization mechanisms. A path expression [3] is used to present the desired design specifications. (Path expressions are primarily used to specify the order in which operations can occur on shared data objects). A formalism is introduced to automatically verify the synchronization mechanism; this formalism is based on a modified Petri net [17] and the vector addition system [22]. Present status of this project is that the techniques are ready to be implemented in software.

---

[4]SDLC is an acronym for Synchronous Data Link Control, used by IBM.

List of Publications

"The SLIDE Simulator: A Facility for the Design and Analysis of Computer Interconnections", A. Altman and Alice Parker, submitted to the 7th Annual Symposium on Computer Architecture, September 1979.

"The GLIDE/SLIDE Compiler", John Wallace, Technical Report, Department of Electrical Engineering, Carnegie-Mellon University, July 1979.

"On Automatic Verification of SLIDE Descriptions", John Wallace, M.S. Project Report, Department of Electrical Engineering, Carnegie-Mellon University, August 1979.

"The CMU Design Automation System: An Example of Automated Data Path Design", Alice Parker, D. Thomas, et al., Proceedings, 16th Design Automation Conference, June 1979.

"SLIDE: An I/O Hardware Descriptive Language", John Wallace and Alice Parker, Proceedings of 1979 International Symposium on Computer Hardware Descriptive Languages, October 1979.

"The Development of GLIDE: A Hardware Descriptive Language", Alice Parker and John Wallace, *Submitted to IEEE Transactions on Computers, being revised.*

"Automated Synthesis of Digital Hardware", Alice Parker and L. Hafer, Submitted to IEEE Transactions on Computers, being revised.

Participating Scientific Personnel

- Alice C. Parker, Principal Investigator
- Donald E. Thomas, Associate Invesigator ✓
- Lui Sha, Graduate Assistant (August 1978)
- Richard Cloutier, Graduate Assistant (September 1978 - August 1979)

References

[1]
T. Agerwala.
Microprogram Optimization: A Survey.
IEEE Transactions on Computers C-25, October, 1976.

[2]
Arthur Altman and A.C. Parker.
The SLIDE Simulator, A Facility for the Design and Analysis of Computer Interconnections.
Technical Report, Carnegie-Mellon University, October, 1979.

[3]
Sten Andler.
Predicate Path Expressions: A High-Level Synchronization Mechanism.
PhD thesis, Carnegie-Mellon University, 1979.
Computer Science Department.

[4]
ANSI Technical Committee X3T9.
U.S.A. Contribution to ISC TC97/SC13 for a Small Computer-to-Peripheral Bus Interface Standard.
December,
1978.

[5]
M.R. Barbacci, W.E. Burr, S.H. Fuller and D.P. Siewiorek.
Evaluation of Alternative Computer Architectures.
Technical Report, Carnegie-Mellon University, 1977.
Department of Computer Science Technical Report.

[6]
M.R. Barbacci, et al.
The ISPS Computer Description Language.
Technical Report, Carnegie-Mellon University, 1977.
Department of Computer Science Technical Report.

[7]
W. Burr, A. Parker, D. Allison, B. Cooper and C. Silio.
Considerations in the Design of a Bus System for the Military Computer Family.
Computer , 1979.

[8]
S. Dasgupta and J. Tartar.
Automatic Identification of Maximal Parallelism in Straight Line Microprograms.
IEEE Transactions on Computers C-25, October, 1976.

[9]
E. DeBenedictis.
Multilevel Simulator.
Department of Electrical Engineering, Carnegie-Mellon University.
M.S. Project.

[10]
D. Dewitt.
A Machine Independent Approach to the Production of Horizontal Microcode.
PhD thesis, University of Michigan, Ann Arbor, 1976.

[11]
M. Gonzales, Jr.
Workshop Report on the Science of Design.
Technical Report, University of Texas, San Antonio, March, 1978.

[12]
L. Hafer and A. Parker.
Register-Transfer Level Digital Design Automation: The Allocation Process.
Proc. of the 15th Design Automation Conference , June, 1978.

[13]
P. Mallett.
Methods of Compacting Microprograms.
PhD thesis, University of Southwestern Louisiana, December, 1978.
Department of Computer Science.

[14]
Michael McFarland.
The Value Trace: A Data Base for Automated Digital Design.
Electrical Engineering Department, Carnegie-Mellon University.
M.S. Project.

[15]
A. Nagle.
Automatic Design of Sequencers for the Control of Digital Hardwar.
January,
1978.
Ph.D. Thesis Proposal, Dept. of Electrical Engineering, Carnegie-Mellon University.

[16]
A.C. Parker, et.al.
The CMU Design Automation System.
In *Design Automation Conference Proceedings No. 16.* ACM SIGDA, IEEE Tech. Comm. on Design Automation, June, 1979.

[17]
C.A. Petri.
Fundamentals of a Theory of Asnchronous Information Flow.
in Proceedings of IFIP, pages 166-168, IFIP, 1962.

[18]
E. Snow.
Automation of Module Set Independent Register Transfer Level Design.
PhD thesis, Carnegie-Mellon University, April, 1978.
Electrical Engineering Department.

[19]
E. Snow, D. Siewiorek and D. Thomas.
A Technology-Relative Computer Aided Design System: Abstract Representations, Transformations and Design Tradeoffs.
In *Design Automation Conference Proceedings No. 15*, pages 220-226. ACM SIGDA, IEEE Comp. Soc. Tech. Com. on Design Automation, June, 1978.

[20]
Carl Sunshine.
Formal Techniques for Protocol Specification and Verification.
Computer 12(9), September, 1979.

[21]
D. Thomas.
The Design and Analysis of an Automated Design Style Selector.
PhD thesis, Carnegie-Mellon University, April, 1977.
Electrical Engineering Department.

[22]
Jan van Leeuwen.
A Partial Solution to the Reachability-Problem for Vector-Addition Systems.
In *6th Annual Symposium on Theory of Computing.* ACM, May, 1974.

[23]
J. Wallace and A. Parker.
SLIDE: An I/O Hardware Descriptive Language.
to be published in the Proceedings of the 1979 International Symposium on Hardware Descriptive Languages, Palo Alto, CA., 1979.

[24]
J. Wallace.
SIGNALS; A Proposed Extension to GLIDE.
Research note, Electrical Engineering Dept., Carnegie-Mellon University, February 1979.

[25]
J. Wallace.
The GLIDE/SLIDE Compiler.
Research note, Electrical Engineering Dept., Carnegie-Mellon University, April 1979.

[26]
C.H. West.
General Technique for Communications Protocol Validation.
IBM Journal of Research and Development 22(4), July, 1978.

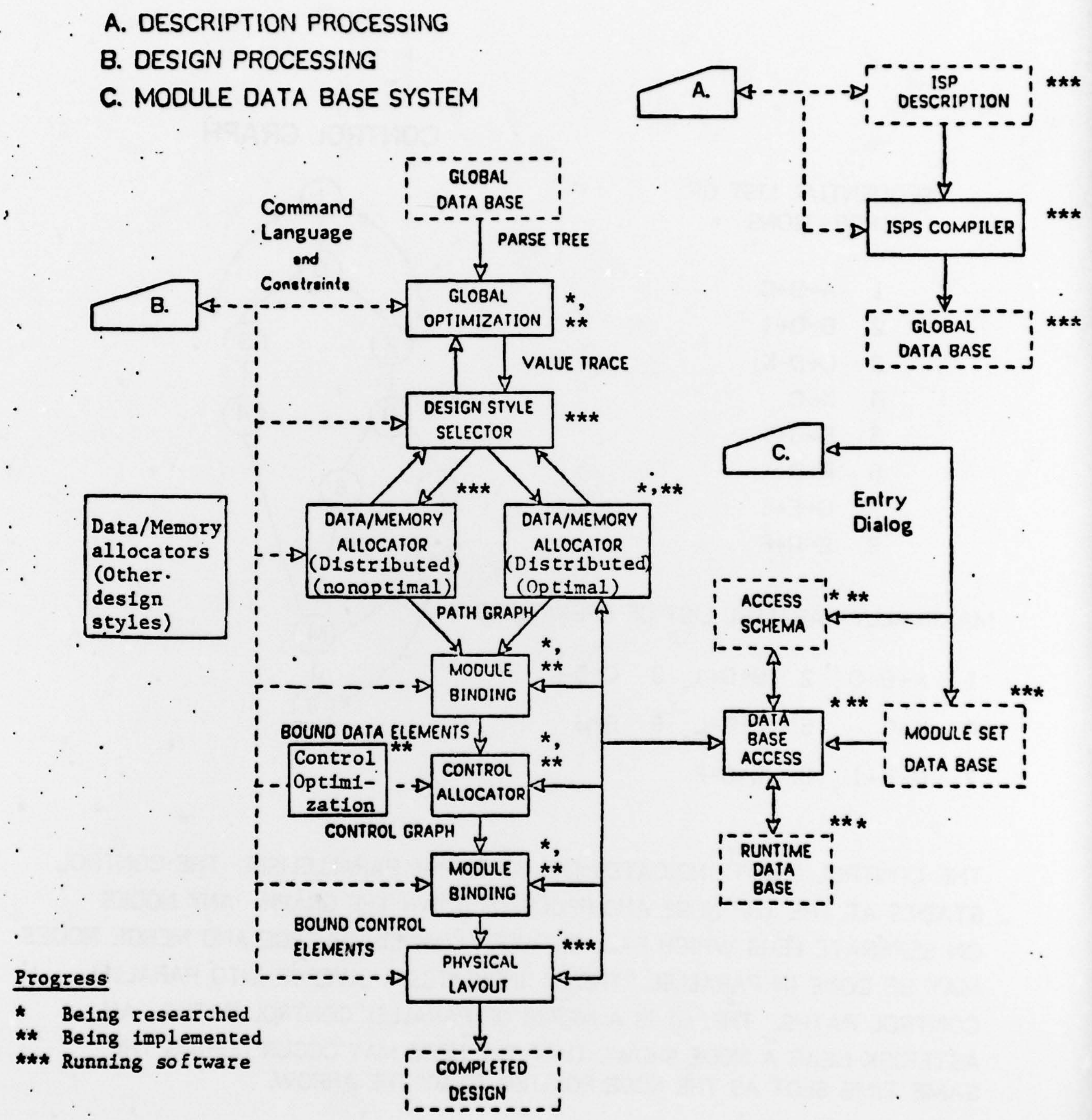


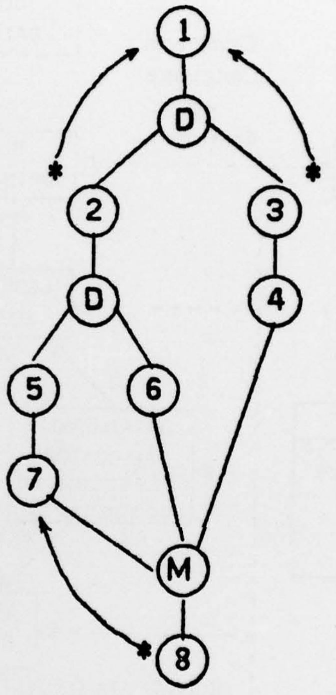
Fig. 1 CMU DESIGN SYSTEM OVERVIEW

Note: Presently used physical layout software was written elsewhere.

CONTROL GRAPH

SEQUENTIAL LIST OF OPERATIONS

1	A←B+C
2	B←D+1
3	C←D-K
4	K←C
5	E←B+L
6	F←B
7	G←E+1
8	E←D+F



MAXIMALLY PARALLEL LIST OF OPERATIONS

1 A←B+C, 2 B←D+1, 3 C←D-K
4 K←C , 5 E←B+L, 6 F←B
7 G←E+1 , 8 E←D+F

THE CONTROL GRAPH INDICATES THE POTENTIAL PARALLELISM. THE CONTROL STARTS AT THE TOP NODE AND PROCEEDS DOWN THE GRAPH. ANY NODES ON SEPERATE LEGS WHICH FALL BETWEEN COMMON DIVERGE AND MERGE NODES MAY BE DONE IN PARALLEL. THE Ⓓ INDICATES A DIVERGE INTO PARALLEL CONTROL PATHS. THE Ⓜ IS A MERGE OF PARALLEL CONTROL PATHS. AN ASTERISK NEAR A NODE SHOWS THAT THE NODE MAY OCCUR DURING THE SAME TIME SLOT AS THE NODE POINTED TO BY THE ARROW.

Fig. 2 The Control Graph Generated by the CMU-DA System.